AcroTeX.Net

The thorshammer Package

D. P. Story

Prepared: January 16, 2020

www.acrotex.net
Version 1.5.7, 2020/01/13

The thorshammer Package

Table of Contents

1 Acknowledgements 3

2 Introduction 3

3 Preliminaries 4
- 3.1 Package requirements . . . 4
- 3.2 Package options . . . 5

4 Necessary conditions that would make this system work 6
- 4.1 The effectiveness of using PDF quizzes for assessment . . . 6
- 4.2 Authoring a quiz . . . 7

5 Workflows for interactive assessment using PDF 8
- 5.1 Basic methods . . . 8
  - Preamble . . . 8
  - The body of the document . . . 11
- 5.2 Quizzes that specify the `useclass` option . . . 16
  - Preamble . . . 16
  - The insertion of class information . . . 18
  - The `\sadQuizzes` command . . . 21
- 5.3 The `usebatch` option: single rendition source . . . 22
  - How to use the `Thor's way` action . . . 22
- 5.4 The `usebatch` option: multiple renditions source . . . 25
  - Preamble . . . 26
  - The body of the document . . . 26
  - The `\sadQuizzes` command (revisited) . . . 27
- 5.5 Quizzes that specify the `batchdistr` option . . . 28

6 Bells and whistles 29
- 6.1 Running Headers . . . 29
- 6.2 Solution pages . . . 30
- 6.3 Cover pages . . . 30
- 6.4 Switches to control program flow . . . 30
- 6.5 Language localization . . . 31
- 6.6 System scripts . . . 32

7 List of sample files 32

8 Summary of action sequences provided 33

9 Further discussion of Basic Methods 34

10 A final note on Thor's workflow 36

11 The `ordinary` option 37

12 My retirement 37

1. Acknowledgements

The author would like to acknowledge Thorsten G. (a.k.a., Thor) who proposed this workflow and who contributed many ideas, the proposed workflow I found to be interesting and worth my time developing the idea; and to Jürgen G. (a.k.a., Loki) who also contributed many good ideas, enthusiasm, questions, bug detection, and motivation. High regards and respect to both.

D. P. Story (a.k.a., Odon)

Warning! The workflow of the thorshammer package requires the instructor to use **Adobe Acrobat XI**, or later. Any PDF creator may be used to build the quizzes, however, but Acrobat is needed to execute one-time JavaScript (\sadQuizzes) and to run the action sequences (formerly called batch sequence by Adobe) such as Thor's way. Only Adobe Reader is required for the students. Of course, this is the full-featured Adobe Reader of a desktop or laptop, not the Adobe Reader found on tablets, smartphones, and such.

2. Introduction

Thor has asked me to assist him in creating a quiz system, based on AcroTeX, to be delivered to his classes. His workflow for this assessment system is as follows:[1]

1. The quiz environment is used to pose the questions, which consist of MC, numerical fill-in the blank, and *extended response questions*.[2] Though the quiz environment is used, the score is not reported to the student upon finishing the quiz.
2. The students takes the quiz in a computer lab, each student has his own personal student folder. The quizzes are dropped into the personal folders.
3. When the student finishes the exam, taken in AR,[3] she presses the End Quiz control and saves the document.
4. At some point, the instructor's script moves the student quizzes to the instructor's folder.
5. The instructor opens the PDF and finishes marking the extended response questions and assigns a grade.
6. System scripts then returns the quizzes to the students.

The original concept was expanded considerably as the package was developed: extensive system scripts were written to support the system; and action sequences were written, also to support Thor's way.

[1] As my occasional friend Jürgen says, this workflow is a real hammer, so I titled this package 'Thors(ten) hammer', or simply thorshammer.

[2] Extended response questions are the interesting part.

[3] AR refers to Adobe Reader DC.

3. Preliminaries

Most important is the correct installation of thorshammer along with its required packages and folder JavaScript files.

3.1. Package requirements

The most recent version of the packages insdljs (2019/10/23), exerquiz (2019/10/13), eq-save (2019/08/07) are required. These three packages were modified slightly to obtain features needed by thorshammer.

aeb_pro.js
aeb-reader.js
folder-js folder

The thorshammer package ships with two folder JavaScript files aeb_pro.js (Version 1.7.2 or later, required) and aeb-reader.js (Version 1.0 or later); these two are found in folder-js folder. The latter is a new JavaScript file, the former ships with the aeb_pro package. These files give access to security restricted JavaScript methods. If you already have aeb_pro installed, be sure you have Version 1.7.2 of aeb_pro.js, if not, install Version 1.7.2 provided by this distribution.

The aeb_pro.js is used on the instructor's system, along with Acrobat (AA), to author quizzes. Distiller is not used unless the author prefers a TEX → DVI → PS → PDF workflow, using Distiller as the PDF creator. For this workflow, pdflatex, lualatex, and xelatex will work as PDF creators; however, Acrobat ia needed to execute the JavaScript code generated by the command \sadQuizzes and to execute the action sequences provided by this package.

Once the quizzes are created, the author can take the quizzes in Adobe Reader (AR) to get the same experience as his students. Additionally, aeb-reader.js can be installed on the instructor's system and on the student's work environment, if possible/permitted. aeb-reader.js contains a subset of JavaScript methods taken from aeb_pro.js that *will enhance student experience*. The author can install this file on his system to take the quizzes in the same environment, again using Adobe Reader. **Warning:** Do not install aeb_pro.js in any folder where an Adobe Reader used by students would read it.

Warning

Other enhancements to user experience. Adobe Reader is by default in Protected Mode. Then Protected Mode is enabled, one or more security dialog boxes popup as the student saves his/her document. To eliminate this annoyance, clear the Enable Protected Mode at startup checkbox, as shown in the Figure 1. On your personal system, open AR, press Ctrl+K, select Security (Enhanced) from the Categories panel, finally, clear the checkbox, as shown in Figure 1.[4] The installation of aeb-reader.js and clearing the checkbox to disable Protected Mode most likely require the sysadmin to make those changes.

Ctrl+K

sysadmin permission

Installation of JS files for Acrobat. The JavaScript file aeb-reader.js is provided in the folder-js folder. Discussion of how and where to install folder JavaScript files is found in the file docs/install_jsfiles.pdf. Read and follow the directions carefully.

Installation of JS files for Adobe Reader. The folder JavaScript file aeb-reader.js may be installed for use by Adobe Reader as well. Installing aeb-reader.js enhances

[4] For AA, this checkbox is cleared by default.

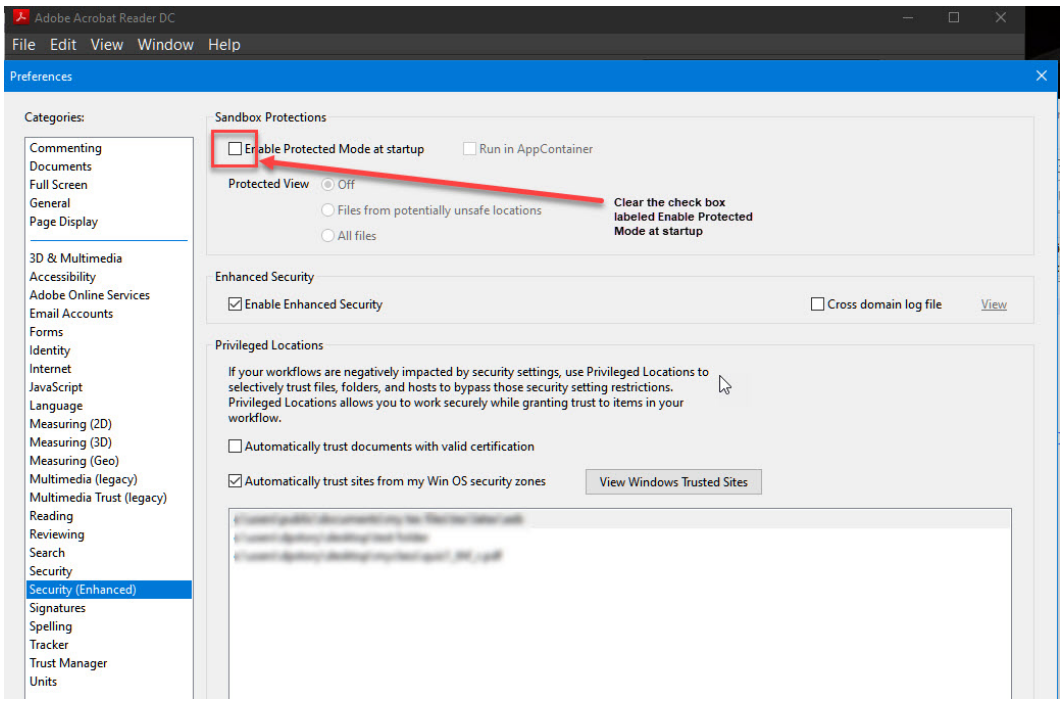


Figure 1: Clear Protected Mode checkbox

the experience of the students as they save their quiz document after completing the quiz. Normally, it is not possible to install such a JavaScript file for use by students; however, if the students take the quiz in a Computer Lab, the sysadmin can install `aeb-reader.js` on all the computers used in the Computer Lab. Discussion of how and where to install folder JavaScript files is found in the file `docs/install_jsfiles.pdf`. Read and follow the directions carefully. (Reader looks for its folder JavaScript files in the same location where Acrobat would look for them.)

3.2. Package options

There are seven options for this package.

`nocfg` There is a language localization file, discussed in Section 6.5, that is loaded as a matter of course. When this option is taken, the configuration file is not loaded. In this case, the default English language definitions are used for all strings.

`testmode` When this option is specified, the quizzes are regular quizzes that can be tested in the usual exerquiz-way. A Correct button is available to correct the quiz.

`!testmode` A convenience option for putting the document into the default mode: Quizzes are ready to be taken by the students, pressing the End Quiz does not give any score for their effort, but students are prompted to save the quiz.

`useclass` Use this option to bring in additional code to declare each member of the class, to automatically build a custom quiz for each class member, to distribute these quizzes to a designated folder of the instructor, and to distribute the quizzes the respective class folder. Refer to Section 5.2 for details.

`usebatch` This option executes the `useclass` option; however, there is a batch sequence, called Thor's way, that the instructor uses to process all class quizzes after the class has finished with them. Refer to Section 5.3 for details.

`batchdistr` The option automatically passes the `usebatch` and the `useclass` options. Additionally, the document assembly script does not distribute copies of the quizzes to the student folders, rather, it is expected the author uses either Thor distributes or Thor protects and distributes action wizards to perform the distribution task. The quizzes can later be distributed using an operating system script. Refer to Section 5.5 for details.

`ordinary` When this option is passed, ordinary quizzes are created, *ones that do not follow Thor's Hammer workflow*. The `testmode` option is automatically executed, as well as other changes. Refer to section 11 for more details and examples.

4. Necessary conditions that would make this system work

In recent years, browser technology has *devolved* to the point that Adobe Reader is no longer supported—with the exception of Microsoft's Internet Explorer 11—within a browser window. For documents using exerquiz, which relies heavily on Acrobat JavaScript API, interactive quizzes can no longer be taken within a brower; hence, if an instructor wants to use exerquiz to author quizzes to be taken by his students, the quizzes must be taken on a desktop or laptop from within the Adobe Reader application.[5]

4.1. The effectiveness of using PDF quizzes for assessment

The effectiveness of an in-class quiz is closely tied to quiz security. In a normal classroom setting, students take the quiz/test—all at the same time—under the glaring gaze of responsible adult. This is to ensure there is no 'cheating.' This is the method used for paper quizzes/tests that are a major part of the students grade. However, security becomes more lax when students work on assignments with lesser weight in the final grade. Homework assignments are an example.

For PDF quizzes, the philosophy is the same, for lesser credit, the students can take the quizzes on their own. If the quizzes are for major credit in the course, then one would expect stricter security.

The ideal condition. The following list comprises the ideal conditions for assessment using PDF quizzes.

[5] At this time, Adobe Reader on mobile and tablets do not support form fields and JavaScript.

- **An exam period.** There is a definite (time) period the students are to take the quizzes. The quizzes are moved to the private student folders at the beginning of the assessment period, and removed again at the end of the assessment.
- The ideal condition is for the students to take the quiz only in a CBT lab (computer-based testing lab) under supervision.
- **CBT Lab.** The student must use Adobe Reader, must not use email or a thumb-drive (to transmit the PDF quiz), and must not print the quiz. For Thor's workflow, the student finds the quiz in the his/her personal folder.
- **No access to private folder.** The students do not have access to the private folder outside of the CBT lab.

The workflows devised in this package will work well for class quizzes taken under these ideal conditions.

4.2. Authoring a quiz

Acrobat required

The thorshammer package uses post-pdf creation JavaScript methods. For these methods to have any effect, the document author must use the Adobe Acrobat application. The PDF creation can be any of the usual LaTeX workflows: dvips/Adobe Distiller, pdflatex, lualatex, or xelatex. In all cases, after PDF creation, the newly created file must be opened in Acrobat before any auxiliary files are deleted. When opened in Acrobat the first time, certain code lines of JavaScript execute to perform a number of tasks.

edit config.js

There is another JS file that needs to be either located or created, the file name is `config.js`. It is a standard Acrobat file and is located in the same folder in which `aeb-reader.js` (or, perhaps, `aeb_pro.js`) reside. The file `install_jsfiles.pdf`, found in the `docs` folder, discusses where Acrobat expects to find folder JavaScript files. Find or create `config.js` and insert the following line:

```
var _thorshammer=true;   // secret variable
```

This JavaScript variable plays an interesting role: Certain form elements in the quiz only appear when this variable is present, in particular, the Mark it control only appears when the document is opened in Acrobat and the above script line appears in the `config.js` file.[6] You'll have to see it to believe it.

Once the quiz has been built, the quizzes themselves require only Adobe Reader to take.

[6] **Important!** Do not install `config.js` where any AR used by a student; otherwise, the Mark It control will be displayed to the student, which we don't want.

5. Workflows for interactive assessment using PDF

The document author has several choices for student/instructor experience. We discuss these in this section.

5.1. Basic methods

basic method

A *basic method* is one in which none of the "class options" are used; these are `useclass`, `usebatch`, and `batchdistr`. This method is illustrated in sample file `thexb.tex`. Compiling this file produces a single quiz, which is distributed to the class.

- **Preamble**

```
\documentclass{article}
\usepackage{amstext}                % (optional, used in a minor way below)
\usepackage{web}                    % (optional)
\usepackage[usealtadobe]{insdljs}
\usepackage[usesumrytbls]{exerquiz}
\usepackage{thorshammer}            % (basic methods imply no options taken)

\setInitMag{fitwidth}                  % (optional, defined in this package)
\hypersetup{pdfpagelayout=OneColumn} % (optional, from hyperref )

% Optional customizations
%\useNameToCustomize
%\enumQuizzes{3}
%\instrPath{/C/Users/⟨username⟩/Desktop/Test Folder/target/_Thor}
%\classPath{/C/Users/⟨username⟩/Desktop/Test Folder/target/myClass}
%\distrQuizzes{{A/_Thor}{B/_Thor}{C/_Thor}}

\reversemarginpar
\showCreditMarkup   % always include this command, it is required
%\previewOn\pmpvOn
\useBeginQuizButton[\CA{Begin}]
\useEndQuizButton[\CA{End}]
\PTsHook{($\eqPTs^{\text{pts}}$)}
\useMCCircles

% Declare the quiz name
\DeclareQuiz{q1}
% Post document creation JavaScript
\begin{docassembly}
\sadQuizzes
\end{docassembly}
\begin{document}
...
```

There are two new commands and one new environment above to mention:

```
\setInitMag{fitpage|actualsize|fitwidth|fitheight|fitvisible|inheritzoom}
```

This command determines the initial magnification. There are a choice of six values for the argument; the default is `fitpage`.

Optional customizations. There are a several commands to customize the creation of the quizzes.

```
\useNameToCustomize
\enumQuizzes{⟨num⟩}
\instrPath{⟨path⟩}
\classPath{⟨path⟩}
\distrQuizzes{{*⟨path1⟩}{*⟨path2⟩}...{*⟨pathnum⟩}}
```

When none of these commands appear in the preamble, compiling the quiz document, produces a single quiz; e.g., compiling `thexb.tex` produces `thexb.pdf`. Otherwise, compiling `thexb.tex` produces, for example, enumerated copies of the quiz files: `thexb-1.pdf`, `thexb-2.pdf`, ..., `thexb-`*num*`.pdf`.

`\useNameToCustomize` When the instructor makes the final assessment, she presses the Mark It button and a Save As dialog box opens: by default, the name of the current file is pre-filled into the File name input field; if `\useNameToCustomize` is in force, however, when Mark It is pressed, the suggested file name in the Save As dialog box is pre-filled as `\jobname-`⟨*first*⟩`_`⟨*last*⟩`-g.pdf`, where `\jobname` is the base name of the original LaTeX source file. The values of ⟨*first*⟩ and ⟨*last*⟩ are taken from the name fields of the document, where the student is expected to enter his name.

`\enumQuizzes{`⟨*num*⟩`}` When this command is present in the preamble, the command `\sadQuizzes` (described below) creates ⟨*num*⟩ copies of the quiz, and labels them `\jobname-1.pdf`, `\jobname-2.pdf`, ..., `\jobname-`*num*`.pdf`.

`\instrPath{`⟨*path*⟩`}` (The instructor folder) The `\instrPath` command is the path to the folder of the instructor. A copy of all quizzes produced are placed at the end of this path.

`\classPath{`⟨*path*⟩`}` (The class folder root) The `\classPath` command is the path to the root of the class folders of the students.

`\distrQuizzes{*{`⟨$path_1$⟩`}{*`⟨$path_2$⟩`}...{*`⟨$path_{num}$⟩`}}` The argument takes a series of paths: (1) if the star-option (*) is *not present*, then ⟨*path*⟩ is the relative path to the student folder, relative to the class folder root; (2) if the star-option (*) is present, the ⟨*path*⟩ is the full path to the student folder. Each folder path is enclosed in braces ({}).

```
\classPath{/C/Users/dpstory/Desktop/Test Folder/target/myClass}
\newcommand{\altClassPath}
  {/C/Users/dpstory/Desktop/Test Folder/target/myOtherClass}
\distrQuizzes{{A/_Thor}{*\altClassPath/B/_Thor}{C/_Thor}}
```

Quizzes are placed (by `\sadQuizzes`) in the student folders:

```
/C/Users/dpstory/Desktop/Test Folder/target/myClass/A/_Thor}
/C/Users/dpstory/Desktop/Test Folder/target/myOtherClass/B/_Thor}
/C/Users/dpstory/Desktop/Test Folder/target/myClass/C/_Thor}
```

When the \classPath/\distrQuizzes combination is used, the \enumQuizzes command is ignored. The number of quizzes to be created is determined by the number of folder paths listed in \distrQuizzes.

Declaring quiz name. Also in preamble is the declaration of the quiz name.

```
\DeclareQuiz{⟨qz-name⟩}   % defined in exerquiz
```

The ⟨*qz-name*⟩ should consist of alpha-numeric characters only (eg, quiz1), and no umlauts, Loki![7] The command saves the quiz name in several ways:

- As equivalent text macros, \thisQuiz and \currQuiz. Either of these of these two must be used to reference the quiz name when you set up the quiz environment; for example,

```
\DeclareQuiz{q1}
...
\begin{document}
...
\begin{quiz*}{\thisQuiz}   % or \begin{quiz*}{\currQuiz}
...
\end{quiz}
```

\thQuizName

- As the text macro \thQuizName. This command was created to solve a problem with the Thor's hammer. (Can you believe it?) When a parent file has multiple renditions of the quiz, Thor's hammer names them differently. For example, if we declare \DeclareQuiz{q1}, in the first rendition, the quiz name is q1a, in the second rendition, the quiz name is q1b, and so on. (There is a limitation of 26 renditions, though the use of alphabetic letters can easily be changed to numbers.) The command \thQuizName always expands the original quiz name.

While on the topic of quiz names, Thor has forged with his mighty hammer, a convenience command,

\thQzName{⟨*text*⟩}	(\theQuizName)	\def\thqzname{⟨*text*⟩}

The expression in the parentheses above is the default value; in the third column is the underlying text macro definition. Declaring \theQzName{Quiz #1} defines the text macro \thqzname that expands to 'Quiz #1'.

[7] I am informed by the mighty Thor himself that umlauts can simply rendered in German with 'ue' instead of ü, 'oe' instead of ö, and so on; however, we don't want to encourage Loki. I wish he would have told Loki years ago, it would have saved me a lot of headaches.

Post document creation JavaScript. It is important to insert the \sadQuizzes command just above \begin{document}, within the docassembly environment.

```
\begin{docassembly}
\sadQuizzes
\end{docassembly}
```

The docassembly environment (also available in the aeb_pro package) is a verbatim write environment; it writes its contents verbatim to the file docassembly.cut then inputs it back in immediately. In this application, place \sadQuizzes in the body of the environment.

What does \sadQuizzes do? The command expands to a series of JavaScript lines. What is does depends on several factors.

- The script identifies the solution pages, extracts and saves them to the current folder under the name \jobname-thsolns4-⟨*qz-name*⟩.pdf; it then removes the solution pages from the parent document. The solution pages are later appended to the quiz after the students take the quiz by the instructor when she presses the Freeze Quiz button.
- If \enumQuizzes{⟨*num*⟩} is used, \sadQuizzes makes *num* copies of the quiz, labeling them \jobname-1.pdf, \jobname-2.pdf, and so on. Files are saved to the folder of the source file.
- If the \classPath/\distrQuizzes combination is used, \sadQuizzes makes enumerated copies of the quiz and places each in a separate folder designated by the argument of \distrQuizzes.

• The body of the document

required Thor *requires* a first and last name field.

```
\FirstName[⟨opts⟩]{⟨wd⟩}{⟨ht⟩}
\LastName[⟨opts⟩]{⟨wd⟩}{⟨ht⟩}
```

The ⟨*opts*⟩ argument is for changing the appearance of these text fields. These fields are placed at the top of the quiz.

```
\FullName[⟨opts⟩]{⟨wd⟩}{⟨ht⟩}
\thfullnameFmt{⟨how-to-format⟩}
```

(See examples/misc/thexrt.tex)
(Default: \thfullnameFmt{#1+" "+#2})

The \FirstName and \LastName fields are required; however, the \FullName command can explicitly appear as well, perhaps on the cover page. This command gets its value from the values of the \FirstName and \LastName fields; ⟨*how-to-format*⟩ is a JavaScript string that displays how you want the full name to appear. Within the argument of \thfullnameFmt, #1 is the first name and #2 is the second name. Another

example of this formatting is `\thfullnameFmt{#2+", "+#1}`, here, the full name is presented as last name first, followed by a command, then the first name. These ideas are demonstrated in `examples/misc/thexrt.tex`.

importance of the required name fields

The name fields play a critical role in Thor's way of doing things. In addition to providing text fields for the student to identify himself, the *required* name fields are used to mark the beginning of a quiz page. *The required name fields must appear at the top of the page* and above the beginning of the quiz. Some of the JavaScript of thorshammer interprets the page on which the required name fields appear as the page that marks the beginning of the quiz.

The following fields are an integral part of Thor's way:

```
\markQz[⟨opts⟩]{⟨wd⟩}{⟨ht⟩}
\freezeOrSave[⟨opts⟩]{⟨wd⟩}{⟨ht⟩}
\studentReport[⟨opts⟩]{⟨wd⟩}{⟨ht⟩}
\studentGrade[⟨opts⟩]{⟨wd⟩}{⟨ht⟩}
\begin{sumryTblAux}{\currQuiz}
\displaySumryTbl[⟨opts⟩]{\currQuiz}
\end{sumryTblAux}
```

These commands and environments are aesthetically arranged at the top of the quiz file along with the name fields, though the `\displaySumryTbl` can be placed anywhere (see the exequiz documentation on this summary table structure).

`\markQz` This button appears in Acrobat only when the secret variable is detected. After the students have taken the quiz and they are back in the possession of the instructor, the instructor opens each completed quiz and presses this button, whose default caption is Mark It. The underlying script marks up the quiz, showing which problems were answered correctly, how many points received for each response, etc.

One of the features of Thor's way is to have extended response questions. The student responses to this type of question must be read and corresponding points must be assigned in the box provided.

`\freezeOrSave` This button appears only after the Mark It button has been pressed. It has two different forms:

1. When the `usebatch` option (or higher) is *not in effect*, this button appears with caption Freeze Quiz. The action of this button is to attach any solution pages and make all form fields *in the entire document* readonly. It also flattens all annotations in the document. *Press this button only* after all markups are finished and document is ready to be moved into the student's folder.

   Control of flattening. There are options to have the Freeze Quiz button to flatten or not. `\thQuizHeader` and `\thQuizHeaderLayout`:

   ```
   \flattenOn  \flattenOff
   ```

For basic methods, the default is \flattenOff, while for useclass it is \flattenOn. Normally, when you use the useclass option, all fields are flattened when the Freeze Quiz button is pressed. This means you cannot run the quizzes through Thor's way to extract the grades; recording of grades is done in the tradional way, by hand. However, turning \flattenOff enables you to use Thor's way even with the useclass option. (Or, simple change the option to usebatch.)

Applies to basic methods. The following command is obeyed only when there is no "class option" option specified.

\useNameToCustomize	Obeyed for basic methods only

When the \freezeorSave button is pressed, the Save As dialog is opened to save the file, the file name offered is either \jobname.pdf (the default) or \jobname-⟨*first*⟩_⟨*last*⟩-g.pdf (provided \useNameToCustomize has been expanded in the preamble.

2. When the usebatch option (or higher) option *is in effect*, the caption is Save & Close. As the caption suggests, the action is to save the document the current folder and close the document in Acrobat. With the usebatch option, the implication is that the instructor is to use one of the action-wizards to continue processing the students' quizzes.

\studentReport (text field) This readonly field shows the number of points awarded and the total points (eg, 15 / 20). The field is initially hidden, and becomes visible when Mark It is pressed.

\studentGrade (text field) A field for assigning some sort of grade, which can be a letter or number. The field is initially hidden, and becomes visible when Mark It is pressed.

Speaking of aesthetically pleasing arrangement, thorshammer defines, \thQuizHeader and \thQuizHeaderLayout:

```
\thQuizHeader*
\thQuizHeaderLayout
```

The \thQuizHeader command does nothing more than to test for the presence of the * option; if *not present*, a \newpage command is issued, otherwise, no \newpage is forced. Throughout the sample files, \thQuizHeader is used. The \thQuizHeaderLayout is the command that actually contains the arrangement of the design elements. Do not redefine \thQuizHeader; but you may define \thQuizHeaderLayout; look at thorshammer.dtx for the definition \thQuizHeaderLayout, this will give you insight for any redefinition.

preamble **Placement.** \DeclareQuiz is preferably placed in the preamble and \thQuizHeader is placed prior to the first quiz.

```
...
\DeclareQuiz{⟨qz-name⟩}
...
\begin{document}
...
\thQuizHeader
...
⟨begin-quiz⟩
```

instructions Following the above elements comes some instructions for the students, then an exerquiz quiz environment,

```
\begin{quiz*}{\currQuiz}
Solve each of these problems, passing is 100\%.
\begin{questions}
...
\end{questions}
```

The quiz itself is a standard exerquiz quiz, which, if you are using this package, you should be familiar with. Thor's way, however, incorporates extended response questions into it that must be personally evaluated by the instructor. For extended response questions, exerquiz provides the command \RespBoxEssay. This command has rarely been used. Below is the syntax for this command as well as one support command (\essayQ), contained in a rough quiz outline below.

```
\begin{quiz}{\currQuiz}
...
\begin{questions}
\item ⟨a question⟩
...
\essayQ{⟨nPts⟩}
\item\PTs{⟨nPts⟩} ⟨pose-question⟩\par
      % allow space to respond to the question
      \RespBoxEssay[⟨opts⟩]{⟨wd⟩}{⟨ht⟩}

\essayitem{⟨nPts⟩} ⟨pose-question⟩\par
      % allow space to respond to the question
      \RespBoxEssay[⟨opts⟩]{⟨wd⟩}{⟨ht⟩}
...
\end{questions}
\end{quiz} ...
```

Use the \essayQ command prior to the use of \RespBoxEssay. The argument of \essayQ is the number of points for the question. The number of points is repeated again with the \PTs command.

The \essayQ generates a text field in the margin. During the instructor review phrase, the instructor can award points for the essay problem by entering a numerical value into the text field. The value entered will be figured into the totals displayed by the text field created by \LngPtsFld (discussed below). The (default) width and height of

the \essayQ are \def\EsW{33bp}\def\EsH{14bp}. The \essayitem is a convenience command, it expands to \essayQ{⟨*nPts*⟩}\item\PTs{⟨*nPts*⟩}.

Following the quiz are several form fields.

\completeMsgFld[⟨*opts*⟩]{⟨*wd*⟩}{⟨*ht*⟩}	
\stuSaveBtn[⟨*opts*⟩]{⟨*wd*⟩}{⟨*ht*⟩}	
\ShrtPtsFld[⟨*opts*⟩]{\currQuiz}	wd: \PTFW, ht: \DefaultHeightOfWidget
\LngPtsFld[⟨*opts*⟩]{\currQuiz}	wd: \PTFW, ht: \DefaultHeightOfWidget
\TotalsFld[⟨*opts*⟩]{\currQuiz}	wd: \PTFW, ht: \DefaultHeightOfWidget

These form elements are arranged according your design preferences.

\completeMsgFld Initially hidden, the text field becomes visible when the student presses the End Quiz control. The appearing text field reminds the student to save the quiz.

\stuSaveBtn Initially hidden, the button becomes visible when the student presses the End Quiz control. When the student presses the button, the document is presented with a dialog to save the document. After saving, the document is automatically closed.

\ShrtPtsFld Initially hidden, the text field changes to visible when the instructor presses the Mark It control. This field is the \PointsField command of exerquiz, and contains the number of points award for all non-extended response problems.

\LngPtsFld Initially hidden, the text field becomes visible when the instructor presses the Mark It control. It is a field that will hold the total points awarded by the instructor for extended response questions.

\TotalsFld Initially hidden, the text field becomes visible when the instructor presses the Mark It control. This field holds the total of \ShrtPtsFld and \LngPtsFld.

The above elements may be used directly in the document, but it is easier to use the command \thQuizTrailer:

```
\DeclareQuiz{⟨qz-name⟩}
\begin{document}
\thQuizHeader
\begin{quiz*}{\currQuiz}
⟨Instructions⟩
\begin{questions}
...
\end{questions}
\end{quiz*}\quad\thQuizTrailer
```

Skeleton outline of a quiz
The placement of the \thQuizHeader and \thQuizTrailer commands

which incorporates the design elements described above. Throughout the sample files, bothe \thQuizHeader and \thQuizTrailer are used.

Student's experience. The student is presented with a quiz built from `quiz` environment of the `exerquiz` package. To begin the quiz, he presses the Begin Quiz control. If he has not filled the name fields, an alert box appears to instruct him fill in the name fields. He cannot begin the quiz until that occurs. Once the name fields are filled, the student can once again press the Begin Quiz and is allowed to respond to the questions. On finishing, the student presses the End Quiz control. An alert box appears querying the student whether he truly wants to end the quiz. (Ending the quiz means the student cannot make any changes to his responses without having to start over again.) After the student responds by pressing the Yes control on the alert box, the Save control appears, and the student can press it and open a Save As PDF dialog, which eventually ends with saving the file (hopefully in the student's private folder). After a successful save, the document is immediately closed.

Instructor's experience. When the instructor opens a completed quiz, she presses the Mark It control. The Freeze Quiz control appears; the fields `\completeMsgFld` and `\stuSaveBtn` are hidden; and the fields `\ShrtPtsFld`, `\LngPtsFld`, and `\TotalsFld` become visible. Additionally, the credit mark up fields appear in the margins indicating the number of points the student received for each response. If there are any extended response questions, the instructor enters an evaluation of the student's responses in the `\essayQ` field. Changes in score are reflected in the `\LngPtsFld`, `\TotalsFld`, and in the summary table. To complete her evaluation of the quiz, she enters a value into the `\studentGrade` (this is required, saving the file is not permitted, otherwise) and then presses the Freeze & Save control. The file is saved under the name `\jobname-`⟨*first*⟩`_`⟨*last*⟩, where ⟨*first*⟩ and ⟨*last*⟩ are the values entered into the name fields by the student. The instructor must manually record the student's score.

manually record

That is the simple case! Again, see the sample file `theb.tex`.

Figure 2 on page 17 shows the workflow: (a) The student views and takes the quiz; the screenshot is just before the End control is pressed; (b) shows the same quiz after the instructor has pressed the Mark It control; (c) the instructor has given some points in the extended response question (5 pts) and set the grade as 4; (d) this screenshot shows the quiz after the Save & Close button is pressed.

5.2. Quizzes that specify the `useclass` option

class method

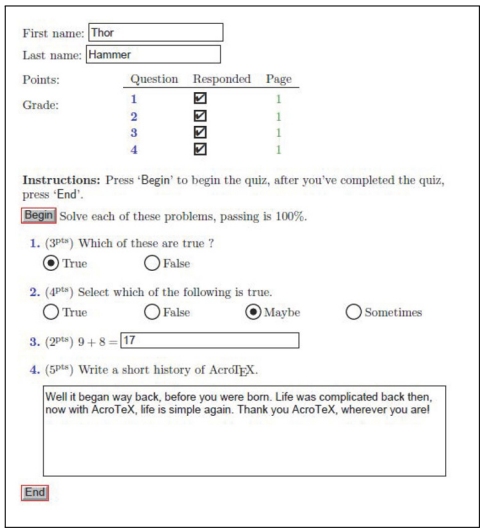
A *class method* is one in which one of the "class options" are used; these are `useclass`, `usebatch`, and `batchdistr`.

The sample file for this section is `thexuc.tex`.

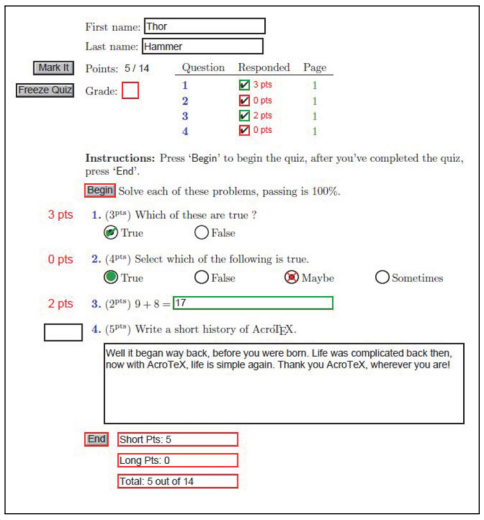
All the elements of the basic methods of Section 5.1 are still critical to the construction of the quiz, additionally, we use the `useclass` option. When this option is taken, it is expected that class information is read into the document, see the discussion below.

- **Preamble**

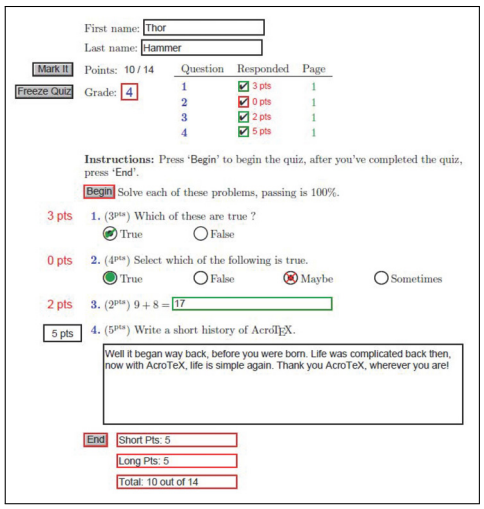
```
\documentclass{article}
\usepackage{amstext}                % (optional, used in a minor way below)
\usepackage{web}                    % (optional)
```



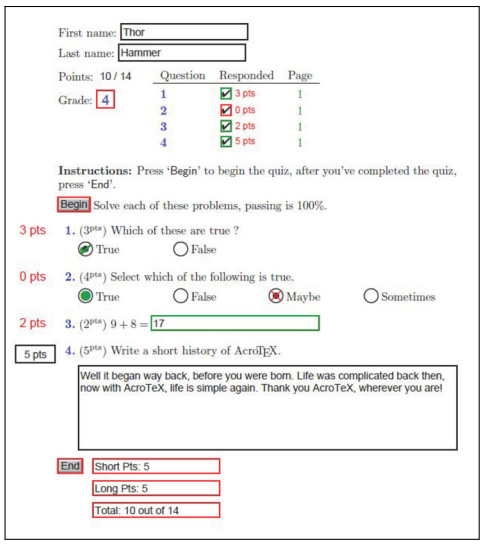
(a) Student's view: quiz taken



(b) Instructor's view: Mark It pressed



(c) Instructor's view: quiz is marked



(d) Student view: quiz returned frozen

Figure 2: Student and Instructor experiences

```
\usepackage[usesumrytbls]{exerquiz}
\usepackage[useclass]{thorshammer}    % (because I am a classy guy)

\setInitMag{fitwidth}                 % (optional, defined in this package)
\hypersetup{pdfpagelayout=OneColumn} % (optional, from hyperref )
\reversemarginpar
\showCreditMarkup
%\previewOn\pmpvOn
\useBeginQuizButton[\CA{Begin}]
\useEndQuizButton[\CA{End}]
\PTsHook{($\eqPTs^{\text{pts}}$)}
\useMCCircles

⟨insertion of class info⟩

% Perform certain tasks when opened in Acrobat the first time
\begin{makeClassFiles}
\sadQuizzes
\end{makeClassFiles}
\begin{document}
...
```

Information on the ⟨*insertion of class info*⟩, on the `makeClassFiles` environment, and on the `\sadQuizzes` command are found in the next two sections.

• The insertion of class information

Thor's way states that quizzes are distributed to student private folders, and copies are saved to the instructor folder.

Required information:

- The path to the instructor folder
- The path to the class folder
- For each student in the class, the first name, last name, and path to the student's folder, relative to the class folder.

These three items are passed to thorshammer through the following commands.

```
\instrPath*{⟨path⟩}
\classPath*{⟨path⟩}
\classMember*{⟨first⟩}{⟨last⟩}{⟨folder⟩}
\classMember*{⟨first⟩}{⟨last⟩}*{⟨full-path⟩}
```

Details of these commands are found next.

`\instrPath` The path to the instructor's folder. Copies of all files are dumped into this folder. The instructor uses this folder to archive the quizzes. The ⟨*path*⟩ may be a full path, which is the default, or a path relative to the folder the source

file is being compiled in. The ⟨*path*⟩ is interpreted as relative to the current path when the *-option is taken; for example:

```
\instrPath{/c/users/thor/documents/myuni/spr19/algebra/myclass}
\instrPath*{myclass}   % assumes source is in the algebra folder
```

`\classPath` The path to the class folder. The ⟨*path*⟩ is an absolute path by default, or a relative path, relative to the source folder. For example,

```
\classPath{/c/users/thor/documents/myuni/spr19/algebra/myclass/staging}
\classPath*{myclass/staging}   % assumes source is in the algebra folder
```

`\classMember` The arguments for this command provide the basic information needed for Thor's way. When the *first* *-option is taken, the three arguments are filtered through `\pdfstringdef` (hyperref). When the *second* *-option is present, the last required argument ⟨*full-path*⟩ must be full path to the student folder.[8]

There needs to be one `\classMember` command for each member of the class.

On the topic of accents. The first and last name arguments may contain accents from a local language. There are several ways of handling them.

- (*-option) `\classMember*{J\"{u}rgen}{Loki}{JL53456/thor}`
- (octal-method) `\classMember{J\oct374rgen}{Loki}{JL53456/thor}`
- (unicode-method) `\classMember{J\u00FCrgen}{Loki}{JL53456/thor}`

There are three methods of introducing the required class information into the quiz document.

Direct placement in the source file. This is the method that messes up your preamble the most:

```
\instrPath{/c/users/thor/documents/myuni/spr19/algebra/myclass}
\classPath{/c/users/thor/documents/myuni/spr19/algebra/myclass/staging}
\classMember{Fred}{Flintstone}{FF34345/thor}
\classMember*{J\"{u}rgen}{Loki}{JL53456/thor}
...
\classMember{Peter}{Pan}{PP75464/thor}
```

If the third argument of the `\classMember` command is empty, the corresponding quiz is placed in the folder determined by the path given in `\classPath`.

The next two methods use a (class) configuration file to load the info.

Use `\InputClassData` to input class info. This is the simplest method. It consists of the direct method cut and pasted into a CFG file.

`\InputClassData{⟨base-name⟩}` % preamble only

This command inputs the configuration file ⟨*base-name*⟩`.cfg`. To repeat, the contents of ⟨*base-name*⟩`.cfg` is simply the lines as displayed in the direct placement method above.

[8] In a perfect world, all student folders are at the end of the (current) class path; however, some students' class folder may be elsewhere, this second *-option is designed for these "exceptional" students.

Use \InputFormattedClass to input class info. This is a more flexible method of inputting class info. The syntax for \InputFormattedClass is,

\InputFormattedClass[⟨\cmd⟩]{⟨base-name⟩} % preamble only

The optional ⟨\cmd⟩ is used to process each class member's info and ultimately builds the \classMember command for that class member. The default for ⟨\cmd⟩ is the command \classMember. The name of the configuration file is ⟨base-name⟩.cfg. The file format is as follow:

```
\instrPath{⟨path⟩}
\classPath{⟨path⟩}
\bClassData
⟨entry1⟩
⟨entry2⟩
...
```

Above the \bClassData marker, direct input is performed, so the \instrPath and \classPath command are placed there. Below the \bClassData marker is the class members individual data, represented here by ⟨entry⟩. When the configuration file is input, ⟨\cmd⟩ is prefixed to each entry:

```
⟨\cmd⟩⟨entry1⟩
⟨\cmd⟩⟨entry2⟩
...
```

A straightforward example is,

```
\instrPath{/c/users/thor/documents/myuni/spr19/algebra/myclass}
\classPath{/c/users/thor/documents/myuni/spr19/algebra/myclass/staging}
\bClassData
{Fred}{Flintstone}{FF34345/thor}
*{J\"{u}rgen}{Loki}{JL53456/thor}
...
{Peter}{Pan}{PP75464/thor}
```

When we expand \InputFormattedClass{algclass}, the lines below \bClassData marker are read in as:

```
\classMember{Fred}{Flintstone}{FF34345/thor}
\classMember*{J\"{u}rgen}{Loki}{JL53456/thor}
...
\classMember{Peter}{Pan}{PP75464/thor}
```

To illustrate the greater flexibility, we offer up this example.

```
\instrPath{/c/users/thor/documents/myuni/spr19/algebra/myclass}
\classPath{/c/users/thor/documents/myuni/spr19/algebra/myclass/staging}
\bClassData
{Fred}{Flintstone}{⟨other-data⟩}{FF34345/thor}
*{J\"{u}rgen}{Loki}{⟨other-data⟩}{JL53456/thor}
...
{Peter}{Pan}{⟨other-data⟩}{PP75464/thor}
```

Notice the entry has an addition piece of information (perhaps a university ID number). Here, each ⟨*entry*⟩ has more information than is required, so we need to extract what we need. For this purpose, we define `\ParseClassMember` and input it as the optional argument of `\InputFormattedClass`.

```
\newcommand\ParseClassMember[4]{\classMember{#1}{#2}{#4}}
\InputFormattedClass[\ParseClassMember]{myalgclass}
```

The first ⟨*entry*⟩ is processed as follows:

```
\ParseClassMember{Fred}{Flintstone}{⟨other-data⟩}{FF34345/thor}
```

which, in turn, expands to `\classMember{Fred}{Flintstone}{FF34345/thor}`.

In theory, ⟨*entry*⟩ can be quite general, but the ⟨*\cmd*⟩ must to be able to parse it and generate the expected data (`\classMember*{`⟨*first*⟩`}{`⟨*last*⟩`}{`⟨*folder*⟩`}`).

• The `\sadQuizzes` command

We now discuss the insertion of the following environment and command:

```
\begin{makeClassFiles}
\sadQuizzes
\end{makeClassFiles}
```

The `makeClassFiles` environment is a simplified version of the `execJS` environment of the `insdljs` package and is similar to `docassembly` introduced earlier. The `\sadQuizzes` command (it does not mean sad news for the students on this quiz) expands to some JavaScript that saves and distributes the quizzes of the students to the designated folders, but `\sadQuizzes` do more than that:

- If solution pages are present, these pages are extracted from the parent document, saved to the instructor's folder, and then deleted from the parent document.
- If there are cover pages, these are removed and saved to the instructor's folder.
- For each member of the class, it builds a customize quiz
  - It reinserts the cover pages, if any.
  - It populates the name fields with the first and last name of the current class member being processed.
  - It saves each (customized) quiz to the appropriate student folder with the file name `\jobname-`⟨*first*⟩`_`⟨*last*⟩`.pdf`
  - It saves a copy of the (customized) quiz to the instructor's designated folder.

Student's experience. Same as **Student's experience**, as described in Section 5.1 on page 16, but the student does not enter his name in the name fields. One of the tasks of `\sadQuizzes` is to pre-populate the name fields with the names of the students.

manually record

Instructor's experience. Same as **Instructor's experience**, as described in Section 5.1on page 16. The instructor must manually record the student's score.

5.3. The usebatch option: single rendition source

The demonstration file for this section is theexub.tex.

rendition discussed

There can be only one quiz (as declared by the \DeclareQuiz{⟨*qz-name*⟩} command); however, there can be more than one *rendition* (or one variation) of the quiz. In this section, we deal with the simplest case: the source file has only one rendition of the quiz. In Section 5.4, the case of multiple renditions is covered.

File construction. A quiz constructed with the usebatch option is the same as one that uses useclass (see Section 5.2); in fact, specifying the usebatch option automatically passes the useclass option to thorshammer. The difference is in the instructor's experience; when the instructor presses the Mark It control, instead of the Freeze Quiz button appearing, the Save & Close button appears. After the instructor is finished marking up the quiz, as described in **Instructor's experience** on page 16, he presses the Save & Close button, which saves and closes the quiz document. After the instructor has inspected each of the quizzes, he is ready to use Thor's way action-wizard (or batch sequence).

• How to use the Thor's way action

A powerful feature of the Acrobat application is the Action Wizard. The user-interface allows you to define a custom action that performs a series of tasks on each PDF in the list of PDFs to be processed. The thorshammer package provides its own action-wizard titled 'Thor's way'.

What does Thor's way do? For each of the quizzes that are designated as the files to be processed, this action-wizard performs the following tasks:

1. If solutions were provided for this quiz, the solution pages are retrieved from the instructor's folder and inserted onto the end of the quiz.
2. Gets the values for the name, student report and student grade fields, and holds them as JavaScript variables.
3. Freezes the document (makes all fields readonly and flattens all pages)
4. Saves the frozen quiz under the name ⟨*filename*⟩-g.pdf. Recall the filename is \jobname-⟨*first*⟩_⟨*last*⟩, where \jobname is the base name of the (parent) LaTeX source file that generated the quizzes to begin with. The file is saved to the folder designated as the 'save folder.'

Additional details are contained in the paragraph **How to use Thor's way** found below.

How to install Thor's way.

1. Start Acrobat
2. select Tools from the toolbar
3. Find and open the Action Wizard
4. Select Manage Actions

5. In the Manage Actions dialog box, press the Import button on the right-hand side
6. Navigate to the `action-wizards` folder of the thorshammer package and choose either Thor's way (en).sequ or Thor's way (de).sequ (both can be installed). The wizard is now installed.
7. Back out of where you are, saving as you go

How to use Thor's way. Use this action after the instructor has finished reviewing each of the quizzes by first pressing the Mark It button, assigning additional points as needed, and pressing the Save & Close button.

Preliminaries

This package comes with two files: `control.pdf` and `terminate-batch.pdf`. *These two files are always placed in the same folder together.*[9] Copy these two files to some folder convenient to the work you are doing.

Preparation phase

1. Open the `container.pdf` file in the Acrobat window, see Figure 3. Purpose of this file is to provide the Thor's way some needed basic information, it also holds the summary of the quiz results as an attachment. The document contains two push buttons, two text fields, and two check boxes.

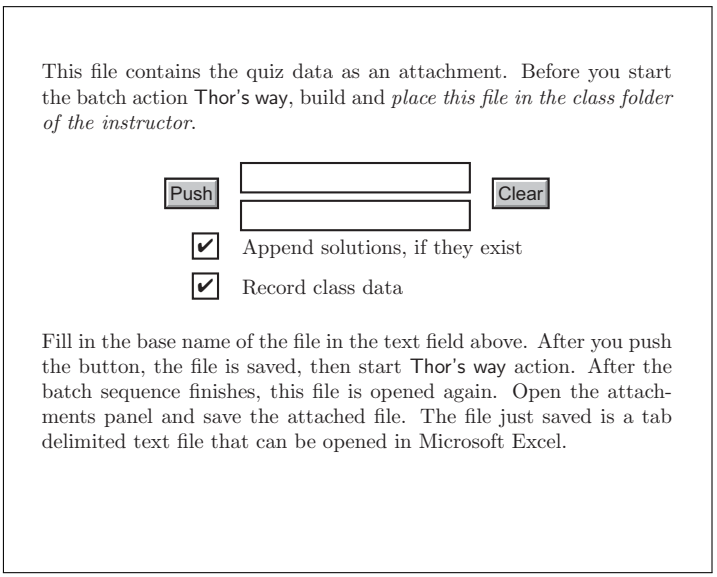


Figure 3: `container.pdf`

[9] Source files for these two PDFs are provided so you can localize any instructions to the land of your choice.

2. With the `container.pdf` file open, in the top most text field, enter a name for the quiz. This will be the base name for the tab-delimited TXT file that is attached to `container.pdf`

3. In the bottom most text field, enter the folder (path) where the quiz files are to be saved. (These are the ones named `\filename-g.pdf`) The folder path you enter can be a relative to the folder that contains `container.pdf`, or it can be a full (absolute) path to the folder. eg,

   ```
   /c/users/thor/documents/myuni/spr19/algebra/myclass/graded
   ```

   Notice the method of referencing the path, this is the device-independent file path reference in the Acrobat JavaScript API Reference manual. See the short article at AcrobatUsers.com for more information.[10]

4. ☑ Append solutions, if they exist
   This check box is checked by default. Clear this box if you don't want any solutions to the quiz appended. The Thor's way action sequence appends the solutions or not according to the state of this checkbox.

5. ☑ Record class data
   This check box is checked by default. If you don't want to record the class data, clear this check box. Normally, this box is always checked; however, if you run Thor's way with Append solutions, if they exist cleared and at a later time you want to append the solutions, run Thor's way again on the same set of quizzes with Record class data cleared. The quiz data file attached to the `container.pdf` will be preserved.

6. Now press the button labeled Push. The information of the text fields are saved as global JavaScript variables that can be accessed by the batch sequence. When you press Push, the file `container.pdf` closes, don't worry.

Action phase

Action: Thor's way

5. Open the Action Wizard and choose Thor's way from the list of actions.

6. **Select files to be processed.** Click on the down arrow control, as indicated in Figure 4. Choose the Add Folders... control if your quizzes are in a folder structure, Acrobat should find all PDFs in the folder (including subfolders) and list them. Choose the Add Files... controls if the quizzes are all in the same folder and you can select them all.

7. **Select `terminate-batch.pdf`.** With the Add Files... control, select the package file `terminate-batch.pdf`, which resides in the same folder as `control.pdf`. This file must be last in the list of files to be processed; if necessary, press the down arrow control and select Manage Files (see Figure 4). The Manage Files dialog box, Figure 5, lists all the files to be processed. Using the down arrow control in the right, move the `terminate-batch.pdf` to it is last in the list.

[10] On Windows, a Windows path seems to work as well: `c:\users\thor\...\graded`

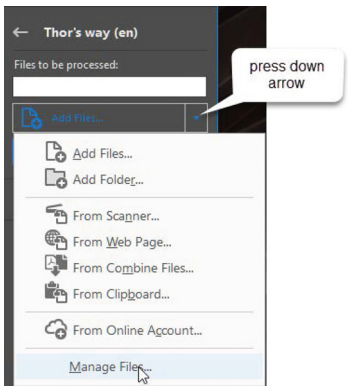


Figure 4: Select files to be processed

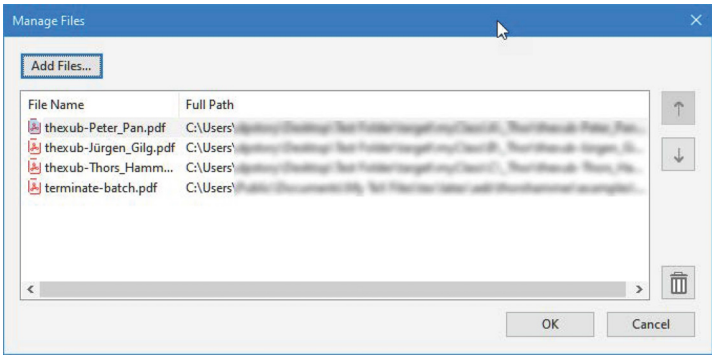


Figure 5: ManageFiles dialog box

8. Keeping your fingers crossed and press the Start control. Each of the quizzes is brought into the Acrobat window, form field information is extracted, frozen, saved, and closed.

9. When the `terminate-batch.pdf` is reached (bottom of the list), no extracting or freezing occurs, but what happens is the `control.pdf` document is opened once again.

10. Look at the Attachments panel, using the user-interface, save the attachment to where you want it. Figure 6 shows the results of a test run of Thor's way. The produced TXT file is a tab-delimited file that can be opened in Microsoft Excel. These saved quiz results can be merged, no doubt, into a parent Excel spreadsheet.

5.4. The `usebatch` option: multiple renditions source

The demonstration files for this section are `thexr.tex` and `thexrt.tex`.

In this section we discuss techniques for introducing multiple renditions of a quiz into

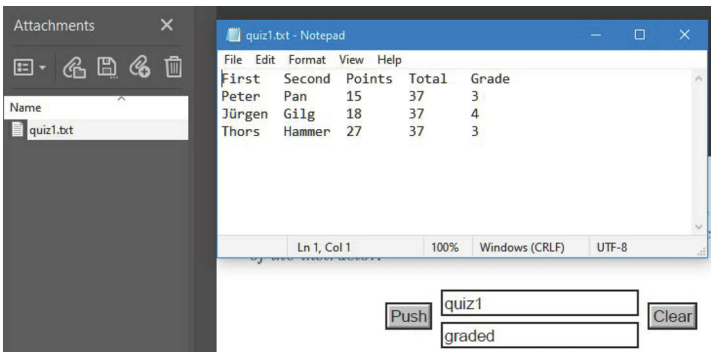


Figure 6: Results displayed

the workflow. To be fair the student, the renditions should be roughly equivalent or the students will cry "foul!" We don't want that to happen, we have enough troubles.

- **Preamble**

Same preamble as previously described, including the use of `\sadQuizzes`.

- **The body of the document**

Following the `\DeclareQuiz` command, declare one or more quiz body names using the `\declareQuizBody` command. Each quiz body declaration has a corresponding quiz body environment that contains a quiz. A rough outline is seen below.

```
% declare the name of this quiz
\DeclareQuiz{⟨quiz-name⟩}

% declare the names of the renditions
\declareQuizBody{⟨qb-name_1⟩}
...
\declareQuizBody{⟨qb-name_k⟩}

% each rendition is a complete quiz within its own environment
% call these environments ''quiz body'' environments
\begin{⟨qb-name_1⟩}
  ⟨quiz-rendition_1⟩
\begin{⟨qb-name_1⟩}
...
\begin{⟨qb-name_k⟩}
  ⟨quiz-rendition_k⟩
\begin{⟨qb-name_k⟩}
```

Actually, there may be only one rendition ($k = 1$), but there is randomness built into the single rendition to produce a different, yet equivalent, version of the quiz.

The ⟨*qb-name*⟩ environments are verbatim environments that write their contents to a CUT file, which are later input back into the document.

Following the ⟨*qb-name*⟩ environments comes the final step, we input these back into the document.

```
\InputQuizBody{⟨qb-name_1⟩}
...
\InputQuizBody{⟨qb-name_k⟩}
```

You can input some or all of the declared quiz bodies, and the quiz bodies may be input several times, especially useful if a quiz body has a degree of randomness built into it.

When you `\DeclareQuizBody`, there is an associated version (or rendition) number in the form of the text macro `\QzVer`, that expands to a number (1, 2, 3, and so on). This number can be incorporated into various section titles, such as in the `\sqslsectitle`, `\thQzHeader`, and `\thQzHeaderCS` commands, for example. Refer to `examples/misc/thexrt.tex` for illustrations.

`\QzVer`

• The `\sadQuizzes` command (revisited)

When you compile a multi-rendition file, and bring the resulting PDF into a PDF viewer, you'll see you have n-quizzes contained in the document, where n is the number of `\InputQuizBody` commands used. Here is a brief description of the action of `\sadQuizzes` in a multi-rendition document.

- If solution pages are present, these pages are extracted from the parent document, saved to the instructor's folder, and then deleted from the parent document.
- If there are cover pages, these are removed and saved to the instructor's folder.
- As the script goes through the list of students, it extracts one of the n quizzes in the parent document and populates it with the name data.
- If there are cover pages for this quiz set, the cover pages are retrieved from the instructor's folder and inserted onto the front of the quiz.
- The extraction of a subset of pages preserves the document JavaScript, but some elements vital to the whole workflow are lost. The script restore these lost elements.
- It saves each (customized) quiz to the appropriate student folder with the file name `\jobname-`⟨*first*⟩`_`⟨*last*⟩`.pdf`
- It saves a copy of the (customized) quiz to the instructor's designated folder.

What you should get is several versions assigned to the class members:

- class member 1 gets the first rendition
- class member 2 gets the second rendition

  ...

- class member n gets the n^{th} rendition, where n = number of `\InputQuizBody` statements
- class member $n + 1$ gets the first rendition

  ...

When the parent document is opened in Acrobat, `\sadQuizzes` executes creates and distributes the quizzes to the instructor's folder and to the student's individual folder. It's just that simple.

Student's experience. Same as their earlier experience, hopefully, this time around it will be a better experience, better results, happier student, we can only hope.

Instructor's experience. Same as above, except she has different quizzes to evaluate within a class. Poor instructor. After she has finished marking the quizzes, she runs them through Thor's way ('How to use the Thor's way action' on page 22), which leads to happiness for the instructor. The student's score is automatically recorded by the Thor's way action sequence.

automatically recorded

5.5. Quizzes that specify the `batchdistr` option

The `batchdistr` option is the same as the `usebatch` option, with only one difference: After the source file is compiled and opened in Acrobat, the newly created quizzes are distributed to the instructor's folder (as declared by `\instrPath`), but *the quizzes are not distributed to the student folders*. The quizzes are distributed to the student's folder using batch sequences, to be detailed in subsequent paragraphs.

Apply security before distributing quizzes. The real purpose of delaying the distribution of the quizzes to the student folders is to place *password security* on the quiz files. After building the instructor's copies of the quizzes, the instructor can use the action Thor protects and distributes. The details of the workflow follows:

1. **Build:** Build the source quiz file using the `batchdistr` option.

   When the source file is opened in Acrobat the first time, the custom-named quiz files are saved into the instructor's folder (as specified by `\instrPath`); no quizzes are deposited in the student folders yet.

2. **Secure and distribute:** With the Acrobat window empty, open the Action Wizard and select the Thor protects and distributes action. For the Files to be processed, select the quizzes that have just been dropped into the instructor's folder. Perform the action by pressing Start.

   Action: Thor protects and distributes

   Note: To delay the distribution of the quizzes, apply Thor protects and later apply Thor distributes.

3. **Examination period:** Now the students take the quiz, the quiz has password security so the student cannot snoop around with the file, even if the student has access to Acrobat.

4. **Marking period:** Following the exam period, the instructor examines each quiz: Press the Mark It button, award any points for extended response questions, assign a student grade, and press the Save & Close button.

5. **Remove security:** Press Ctrl+K to open the Preferences dialog box. In the left-hand panel, select Action Wizard. In the right-hand panel change Security Method, using the dropdown menu to Password Security. Save your changes by pressing the OK control in the lower-right corner.

Action: Thor removes protection

With the Acrobat window empty, open the Action Wizard and select the Thor removes protection action. For the Files to be processed, select the quizzes in the student folders. Use either Add Folder... or Add Files... to select all the quizzes. Perform the action by pressing Start.

6. Press Ctrl+K to open the Preferences dialog box. In the left-hand panel, select Action Wizard. In the right-hand panel change Security Method, using the dropdown menu to Do not ask for password. Save your changes by pressing the OK control in the lower-right corner.

Action: Thor's way

7. **Thor's way:** Now apply the Thor's way action, as described in the paragraph **How to use Thor's way** on page 23.

6. Bells and whistles

6.1. Running Headers

The scheme used here assumes no other LaTeX package has been used to take over the running headers (and footers). If that is the case, use the values of the commands below to design your own.

`\thQzHeaderL{`⟨*text*⟩`}`	(Thor's class)	`\def\th@QzHeaderLQ{`⟨*text*⟩`}`
`\thQzHeaderCQ{`⟨*text*⟩`}`	(Quiz `\thQuizName`)	`\def\th@QzHeaderCQ{`⟨*text*⟩`}`
`\thQzHeaderCS{`⟨*text*⟩`}`	(Solutions: `\thQuizName`)	`\def\th@QzHeaderCS{`⟨*text*⟩`}`
`\thQzHeaderR{`⟨*text*⟩`}`	(`\thepage`)	`\def\t@hQzHeaderR{`⟨*text*⟩`}`

These command declarations allow you to set the left, center (for quiz pages and solution pages), and right running headers. The content of the parentheses to the right are the default values, which obviously need to be changed, except, perhaps, for the right header. The declarations in the first column, define text macros, whose definitions are given in the third column.

The special command `\thQuizName` expands to the name of the ⟨*qz-name*⟩, where ⟨*qz-name*⟩ was declare earlier by `\DeclareQuiz{`⟨*qz-name*⟩`}`. Typically, the choice for the quiz name very simple, such as `q1`. You can have a more meaningful name by declaring `\thQzName{`⟨*friendly-qz-name*⟩`}`; such a declaration defines the text macro `\thqzname` that expands to ⟨*friendly-qz-name*⟩. The package initially declares `\thQzName{\thQuizName}`; for example,

```
\thQzHeaderCQ{Pr\"{u}fung: \thqzname}
\thQzHeaderCS{L\"{o}sungen: \thqzname}
\thQzName{Grammatik 1}
```

\rhPgNumsOnly discussed

The \rhPgNumsOnly command removes all running headers, except for the page number in the right header. Expand in the preamble.

6.2. Solution pages

Thor's hammer does support solutions to questions via the usual `solution` environment. When the parent document is built and opened in Acrobat, the solution pages are identified, extracted, saved to the instructor's folder,[11] and deleted from the parent document. As a result, the quiz the student sees does not contain the solution pages.

6.3. Cover pages

While Thor is hammering away on the solution pages, Thor's hammer also supports the notion cover pages, these are one or more pages appearing in the front of the parent document (prior to any quizzes). These pages are normally ignored when the process of building the custom quizzes for the class; however, if you declare in the *preamble*,

preamble only

```
\DeclareCoverPage{⟨bPg[-ePg]⟩}
```

where ⟨*bPg[-ePg]*⟩ are the beginning (*bPg*) and ending (*ePg*) page numbers; this forms a range of page numbers. As indicated by the syntax, \ePg is optional. The values of these page numbers are 0-based; that is 0 is the first page of the parent document, 1 is the second page, and so on. \DeclareCoverPage{0} states that page 0 is a cover page, while \DeclareCoverPage{0-1} declares the first two pages in the parent document are cover pages. (Page 0 might be a title page, page 1 might be extensive instructions and other information.) Use your knowledge of LaTeX so that there are no page numbers on the cover pages.

0-based numbering

no page numbers on cover pages

6.4. Switches to control program flow

In this section, we provide switches that give control over the behavior the creation and function of a quiz.

```
\autoCopyOn  \autoCopyOff

\instrAutoSaveOn   \instrAutoSaveOff
\instrAutoCloseOn  \instrAutoCloseOff

\stuAutoSaveOn  \stuAutoSaveOff
\stuAutoCloseOn \stuAutoCloseOff

\distrToInstrOn  \distrToInstrOff
\distrToStudentsOn  \distrToStudentsOff
```

[11] For a parent document that uses basic methods (no options), the solutions are saved to the current folder of the parent document.

The defaults for the above commands are the 'On' versions.

`\autoCopyOn` During document development, you don't want to copy the files each time you build and review the document in Acrobat. Set `\autoCopyOff` during quiz development, and declare `\autoCopyOn` when you want to distribute. The default is `\autoCopyOn`.

`\instrAutoSaveOn` When the instructor presses the freeze quiz control, there is an option to automatically save the document or not. `\instrAutoSaveOn` saves the document; however, if `\instrAutoSaveOff` is expanded in the preamble, no automatic save is performed. The default is `\instrAutoSaveOn`.

`\instrAutoCloseOn` When the instructor presses the freeze quiz control, there is an option to silently close the document or not. `\instrAutoCloseOn` closes the document; however, if `\instrAutoCloseOff` is expanded in the preamble, no automatic closing occurs. The default is `\instrAutoCloseOn`.

`\stuAutoSaveOn` When expanded in the preamble, a save button will appear (created by `\stuSaveBtn`) when the End Quiz control is pressed. A dialog appears to save the file, the student can choose the file location and the file name at that time. When `\stuAutoSaveOff` is in effect, the save button does not appear, and the student must press the save button the on Adobe Reader toolbar. The default is `\stuAutoSaveOn`.

`\stuAutoSaveOff` This command is obeyed only if `\stuAutoSaveOn` is in effect. After the student presses the save button (`\stuSaveBtn`), the document is closed after the student save the document. Note that if the student cancels saving the document and if the document still needs saving, the document is not closed. The default is `\stuAutoCloseOn`.

`\distrToInstrOff` Allow the instructor to turn off the distribution of the quizzes to himself by using `\distrToInstrOff`, the default is `\distrToInstrOn`.

`\distrToStudentsOff` This switch allows the instructor to turn off the distribution of the quizzes to the student folders, the default is `\distrToStudentsOn`. (Note: The `batchdistr` option expands `\distrToStudentsOff`.

Except for `\autoCopyOn` and `\autoCopyOff`, there is normally no reason to change these switches from their defaults. They are here to provide additional control.

6.5. Language localization

There are a number of language dependent phrases that appear in the document, these have conveniently been gathered into the file `thorshammer.cfg`. If found on the LaTeX search path, it is automatically loaded, unless the `nocfg` option is in force. This file can be localized to your own language, hopefully no umlauts needed.

6.6. System scripts

In the course of developing this package, several Powershell system scripts were written to illustrate the workflow to ourselves. During the long development of this package, the scripts were folded into a single Powershell script file `thmclass.ps1`. The use of `thmclass.ps1` is described in the file `thmclass.pdf`, found in the `docs` folder.

7. List of sample files

The sample files are found in the `example` folder. Fundamentally, there are only six demonstration files, but these are replicated several times and distributed over several times in sub-folders of the main `examples` folder. The sample files, listed in increasing complexity, and their brief descriptions follow:

`thexb.tex` uses no option and represents the basic methods described in Section 5.1.

`theuc.tex` uses the `useclass` option, as described in Section 5.2.

`thexub.tex` uses the `usebatch` option, as described in Section 5.3. The instructor can use Thor's way action sequence on these quizzes to automatically record the student scores.

`thexbd.tex` uses the `batchdistr` option, as described in Section 5.5. For this file, distribution of the student quizzes is delayed. It is expected to apply some security first.

`ther.tex` uses the `usebatch` option with the `allowrandomize` option for `exerquiz`. As a results, certain MC and MS choices are randomized. The `quizbody` environment encloses the body of the quiz and input back into the document twice. As a result, we obtain two versions of the same quiz, with two randomized order for MC and MS choices.

`theexrt.tex` uses the `usebatch` option with the `allowrandomize` option for `exerquiz` and uses the `ran_toks` package as well. The `ran_toks` package is used to randomize the order of the questions, in addition to the randomization of the choices of `thexr.tex`. The `quizbody` environment is used twice on two distinct set of quiz questions, and input back in several times. The results are three "renditions" of two distinct quizzes, the order of the questions and choices are randomized.

The above files may appear in the following folders.

`nosolns` folder: This is the original collection of demonstration files, all having no solutions associated with any of the questions. (The simplest case.)

`wthsolns` folder: The same six demo files, but with selected solutions to the questions. The `thexb.tex` demonstrates how to set of the document with a screen design (`designi`). It also uses the `spdef` package (provided) to format the document with or without a design option of `web`. Some of the files use running headers.

`cfg` folder: There are several files in this folder, each using the `usebatch` option. They illustrate how to input instructor and student data through CFG files, as described in section entitled 'The insertion of class information' on page 18.

`misc` folder: The `thexrt.tex` file is in this folder. The file has been modified several ways. `\thQuizHeaderLayout` is redefined to hide the `\FirstName` and `\LastName` fields. (Do not delete these two commands, they are required, but we can hide them.) Instead of displaying the first and last name, the `\FullName` command is used on the cover page. The `\FullName` command gets its value from the values of `\FirstName` and `\LastName`. The changes give the quizzes a slightly different look.

`target` folder: The demo files that use the `useclass` option, need a target to drop the instructor's copies of the quizzes and the student copies of the files. In each of the demo files where class information is provided, you need to adjust the paths to taget the subfolders of the `target` folder. The `target` folder may be copied to the desktop, as I did, and target the quiz files there.

8. Summary of action sequences provided

By "actions sequences" we mean batch sequences of Acrobat that perform a series of tasks. Refer the document install-action-seq.pdf for information on how to install action sequences, found in the `docs` folder of this distribution.

- **Thor's way action.** This action is performed on the (student) completed and (instructor) marked quizzes. Refer to **How to use the Thor's way action** on page 22 for more information on Thor's way.
  - **Applies to** quizzes compiled with the `usebatch` or `batchdistr` option.
  - **What it does:** For each quiz, it extracts first name, last name, score, total points from the quiz and saves them to an attachment of `container`; it flattens the quiz and saves the flattened quiz with a suffix of `"-g"` to the folder designated by the `container` file.
  - **Steps to run the Thor's way action**
    - Open the file `container.pdf` (Figure 3) and fill out the fields provided and press the Push button. The file saves the data entered in the form fields then closes.
    - Select the tool Action Wizard, then select Thor's way
    - Files to be processed: Selected the quizzes (using the Add Folder and/or Add Files controls), then select the special file `terminate-batch.pdf`. This latter file *must be the last file to be processed.*
    - Start the batch by pressing the Start control

  After the action is finished, the `control` document should be in the Acrobat window. Check the console window to see if there are any errors. The attachment to this file contains the quiz results, save and do with them what you will, perhaps merging them into a larger spreadsheet.

- **Thor protects action.** Applies to a collection of newly created quizzes that specify the `batchdistr` option. Running this action places security on the quizzes so the student's cannot view the answers even if they are using Acrobat.
- **Thor distributes action.** A general purpose action that returns quizzes from the instructor's folder (or subfolder) to their respective student folders. There are two situations where you would apply this action:
  - The Thor's way action drops the completed and marked quizzes (with suffix `"-g"`) into the folder designated in the `container` file. This action copies them back into the student folders.
  - For a quiz this is compiled with the `batchdistr`, the distribution of the quizzes is delayed (possibly to place security restrictions on them using the Thor protects action). At the time where you want to deploy the quizzes to the students folders, you can use Thor distributes to accomplish that task.
- **Thor protects and distributes action.** Applies to a collection of newly created quizzes that specify the `batchdistr` option. Refer to Section 5.5 for more details. The action is to apply security to the quizzes and then to distribute them to the student folders. This is a combination of Thor protects and Thor distributes.

  Steps to run the Thor protects and distributes action

  - Select the tool Action Wizard, then select Thor protects and distributes
  - Files to be processed: Selected the quizzes (using the Add Folder and/or Add Files controls). These are undistributed quizzes and should reside in the instructor's folder.
  - Start the batch by pressing the Start control.

  After this action is completed, copies of the secured files should now be in the student folders.
- **Thor removes protection action.** This action is a natural continuation of Thor protects and distributes. It removes the security previously placed on the quizzes. This is needed to flatten the quizzes later in the work flow. The whole workflow is described in Section 5.5. In particular, see item 5 titled **Remove Security**. After removing security, we continue with the Thor's way action.

9. Further discussion of Basic Methods

The essential difference between these two methods is how the quiz files are named:

- For basic methods, when multiple quiz files are produced, the naming convention is `\jobname-1.pdf`, `\jobname-2.pdf`, and so on. When only a single quiz file is created, it is named `\jobname.pdf`.
- For class methods, the quizzes are named as `\jobname-`⟨*first*$_1$⟩`_`⟨*last*$_1$⟩`.pdf`, `\jobname-`⟨*first*$_2$⟩`_`⟨*last*$_2$⟩`.pdf`, and so on.

Class methods have three options `useclass`, `usebatch`, and `batchdistr`. It is the `useclass` that distinguishes basic and class methods.

The following are the scenarios for basic methods:

- None of the **Optional customizations** listed on page 9 are specified in the preamble. A single quiz is produced *for each rendition* in the source file. By default flattening is off, so returned quizzes can be run through Thor's way action sequence. The source file can be compiled with `\flattenOff`, in which case manual recording of the grades is needed.
  - Files are created in the source file folder.
  - It is up to the instructor to deliver these quizzes to the class.
  - You can place security on the files using the Thor protects action sequence.
- `\enumQuizzes{`⟨*num*⟩`}`: ⟨*num*⟩ copies of the file is created (with rendition variations if they exist). All files are dropped into the source file folder. Returned files can be run through Thor's way to record results, unless `\flattenOff` is specified in the preamble.
  - Analogous to the `usebatch` option.
  - Files are created in the source file folder.
  - It is up to the instructor to deliver these quizzes to the class.
  - You can place security on the files using the Thor protects action sequence. Quizzes are delivered manually. When the returned quizzes are to be marked, use Thor removes protection followed by the instructor markup step, followed by Thor's way to record the quizzes. This roughly models batchdistr, but without the distribution part.
- `\enumQuizzes{`⟨*num*⟩`}` and `\instrPath{`⟨*path*⟩`}` are specified: ⟨*num*⟩ copies of the file is created (with rendition variations if they exist). Returned files can be run through Thor's way to record results, unless `\flattenOff` is specified in the preamble.
  - Analogous to the `usebatch` option.
  - All files are dropped into the folder determined by `\instrPath`.
  - It is up to the instructor to deliver these quizzes to the class.
  - You can place security on the files using the Thor protects action sequence.
- `\instrPath{`⟨*path*⟩`}`, `\classPath{`⟨*path*⟩`}`, and `\distrQuizzes{`⟨*args*⟩`}`: ⟨*num*⟩ copies of the file is created (with rendition variations if they exist), where ⟨*num*⟩ is the number of folders listed in the argument of `\distrQuizzes`. Returned files can be run through Thor's way to record results, unless `\flattenOff` is specified in the preamble.
  - This combination of command best model the `usebatch` option.

  - A copy of all quizzes are dropped into the folder determined by `\instrPath`.
  - A copy of all quizzes are dropped into the student folders determined by `\classPath/\distrQuizzes`.
  - Specifying `\distrToStudentsOff` in the preamble, you can the place security on the files using the Thor protects and distributes action sequence, which places the security on the quizzes and distributes them to the student folders. This would model the `batchdistr`

- `\classPath{`⟨*path*⟩`}`, and `\distrQuizzes{`⟨*args*⟩`}`: Same as the previous bullet point above.
  - This combination of command best model the `usebatch` option.
  - A copy of all quizzes are dropped into the source folder, since no `\instrPath` is specified.
  - A copy of all quizzes are dropped into the student folders determined by `\classPath/\distrQuizzes`.
  - Specifying `\distrToStudentsOff` in the preamble, you can the place security on the files using the Thor protects and distributes action sequence, which places the security on the quizzes and distributes them to the student folders. This would model the `batchdistr`.

10. A final note on Thor's workflow

There are two computer drives involved in this workflow: (1) a drive on the instructor's work computer designated as ID (instructor's drive); and (2) a system drive that students have access to, designated as SD (system drive).

1. (ID) The development of the quiz occurs on ID, he builds the quizzes (perhaps individualized to the names of the students) and saves them to the student folders on ID, copies are dumped into the instructor's folder on ID.
2. (ID ↦ SD) The system scripts copy the student folders on ID to SD, preserving folder structure.
3. (SD) The class takes the quiz during the quiz period.
4. (SD ↦ ID) After the quiz period, system scripts copy the completed quizzes from the SD student folders to the ID student folders, and deletes the quizzes from SD.
5. (ID) The instructor grades each quiz, awarding points for the extended response questions. Graded quizzes are flattened and marked with a suffix '-g'.
6. (ID ↦ SD) The system scripts returns the graded quizzes to the student folders on SD, where are met with great rejoicing or gnashing of teeth.

11. The ordinary option

This option was prompted by an exerquiz user who had a unique problem: He wanted to put password protection on the Correct button of an exerquiz quiz. He wanted to distribute quizzes to his students. The students would take the quiz, and, on pressing the End Quiz button, would get their score, but the Correct button would be denied to them through a password mechanism.

Solving this problem necessitated some of the special features of thorshammer, without the use of \sadQuizzes.

To see the solution to this problem, go to the examples/orginary-option folder. There are three files in that folder:

- quiz-pwd-to-correct-AA.tex: This file requires the use of Acrobat to create the quizzes. The password is dynamically encoded as the PDF is loaded into Acrobat for the first time.
- quiz-pwd-to-correct-AR.tex: Again, the Correct button is password protected, only Adobe Reader is required to build this file; however, the encrypted password cannot be built dynamically. It must be first computed and hard-wired placed in the source document.
- get-hash-string.tex: This is a companion file to the 'AR' version described above. Build this file using any method. Open it in Adobe Reader and press the Push button. A response dialog box opens, enter your chosen password (PIN number) and press OK. The hash string of your password appears on the text field. Copy the password (or PIN) and the hash string to quiz-pwd-to-correct-AR.tex. Comment out the human readable password in your file and paste the hash string into the value of the _PinCode JavaScript variable, then build the file quiz-pwd-to-correct-AR.tex. Now, the Correct button is password (PIN) protected.

12. My retirement

Now, I simply must get back to it. ☺